Computer Updates

The Newsletter For TS Computer Owners

Volume 3, Number 2 Spring 1986

MID-WEST TS COMPUTERFEST A SMASHING SUCCESS

In early May I packed up my bags and headed for Cincinnati, Ohio to attend the Mid-West TS Computerfest. What a blast it was! The whole affair took on the characteristics of a big party for Timex Owners. Over 250 people turned out for the event. It must have been the biggest gathering of the clan we've ever had. I don't know how other vendors who attended faired, but I did well as did SyncWare News. The festival provided a badly needed shot in the arm for TS fans. I suspect that the enthusiasm displayed there might cause the fest to become an annual event which travels from city to city across the country. Rumor has it another fest is being planned for New York already. It was just a tremendous thrill to see such a concentration of TS paraphernalia and to be able to meet in person with all the vendors (and customers) I have come to know only on the phone or in writing.

PRO/FILE 2068 PLUG-IN CARTRIDGE ALMOST FINISHED

I do not want to make it a habit of filling the pages of Computer Updates with a lot of hype for new products, but in this instance, I am so happy with the way a new cartridge version of Pro/File 2068 is turning out, I simply can't resist telling you about it. With a lot of help from my old pal, P.H. Skipper who has pioneered the use of "cartridge Basic" on the TS2068, and also from another good friend, Bob Fischer, whose "Extensions Newsletter" has given me much food for thought on ideas for improving Pro/File 2068, I have been working for the past few months on a dramatic upgrade of our favorite data base program.

The biggest upgrade is that the program comes on a plug in cartridge which fits under the door in front of the computer. You turn the computer on, and the program comes up running automatically. Because of the nature of cartridge type programs, memory available for data is greatly increased: 37,000 bytes for files! But that's not all. Every enhancement given in the Pro/File book has been added. The fast machine code sort improvement (from the BREAKTHROUGH addendum) is incorporated. The SAVE and LOAD commands work on just the data you have added to decrease cassette time to a minimum. A new MERGE command lets you combine two sets of files into one big one. Printer software is built-in, and you can choose between any of the commercially available interfaces, or the TS2040. If you use a big printer, you can set DEFP to lprint several file lines on just one line of paper (the TV display is still 32 columns however). You can set your TV picture to color, black letters on white paper, or white letters on black paper. A DUPLICATE feature is added which can make copies of records you store. This saves typing time, and provides for the use of FIELD LABELS in a free form data

(goto page 2)

(continued from cover)
structure. You can have as many different Field Label formats as you wish in the same file. Searching has been greatly expanded so as to include OR and NOT separators in addition to the original AND function. Now you can search for Word-1 AND Word-2 NOT Word-3 OR Word 4 AND etc. Even though this program is burned into a chip, you can still add your own Basic or machine code enhancements into RAM. You can flip to your custom mods at will. This includes saving or loading to disk or microdrive. The TALLY feature which performs arithmetic on numbers stored in files has been expanded so as to allow tallys on any desired file line, not just the first line like the original. When adding or editing data, you can lprint the file you're working on without leaving the edit mode. Software has been added which will let you use any IBM compatible keyboard interfaced through the Experimenter's I/O Port if you ever wish to upgrade to a nice keyboard. This software is completely invisible, meaning there are no Pokes, Users, or Gotos necessary to turn the keyboard on or off. The 2068 keyboard works just fine all the time, but if you ever decide to upgrade to an IBM keyboard, all you need to do is connect the keyboard to the port, plug it in, and type! Once its connected, you are not restricted to using the big keyboard only with Pro/File. You can use the IBM keyboard in your own basic programs as well. This is done by setting up a new keyboard channel so any INPUT #4 or INKEY$#4 command will read the IBM keyboard.

There is still time to add new features, so if you have any ideas for making it even better, please send them in. I want to make the program as complete as I possibly can. I anticipate that by the end of June, I will be ready to ship the cartridge Pro/File 2068. The price is set at $59.95 which includes additional instructions for all the new features. I am taking orders now, and to encourage you to order early, I am offering a special "pre-publication sale" which is $20.00 off the regular price. The sale will extend until the finished cartridges are shipped. So if you want to get the new cartridge, and you want to save $20 off the regular price, send me $39.95 plus $1.50 postage right away. I am sure you will be just as pleased with it as I am.

COMPUTER WEATHER STATION: PART 2 TELL WHICH WAY THE WIND BLOWS

In the last issue's "anemometer" article, I showed how to build a simple but effective (and cheap) device to tell how fast the wind blows. Now, I will carry on with plans for an equally important device to determine wind direction. This project, as all future "weather station" articles shall be, is intended to be added onto what has already been covered. It is my hope that by covering the CONCEPTS in this one specific application, you can adapt the same methods in your own completely different projects. So even if you're not the least bit interested in monitoring the weather, perhaps you can obtain something of value for that robot you're building to vacuum the house (now THERE is a marketable product!)

In very simple terms, what we will do in this project is build a weathervane, a very low tech device which has been around for hundreds of years. But we'll add a new twist to this weathervane by hooking it up to the Universal I/O Port so the computer can monitor and record wind direction.

What You Will Need

In keeping with the tradition set in Part 1, of using simple inexpensive commonly available parts, here is a list of most of the items you will need:

1-metal lid to a coffee can or similar small flat piece of metal
1-coat hanger
1-variable resistor (the volume control swiped from a junked radio or TV)
length of hook-up wire
1-555 Timer Chip (Radio Shack)
1- .1uf capacitor
1- 100 Kohm resistor
1- 1 Kohm resistor

In addition, you will need some type of 8 bit parallel I/O port such as my Experimenter's Universal Input/Output Port which is described in my catalog.
The software presented here is designed to work with this port, however any port will do as long as you change the IN instructions to correspond to the address of the port you use.

General Principles

How do you take the twisting motion of a weathervane and convert it into something meaningful to a computer? I tinkered with quite a few ideas before I struck upon the method shown here. I started by building an array of 4 electrical contacts (one for each compass point) which were configured in a circle around a rotating vane. On the vane, I attached metal brushes which would close a circuit which was then fed into the input port. Depending on which circuit was closed at any given time, I could tell which way the vane was pointing. I arranged the brushes and the contacts they touched in such a way that four contacts could even determine Northeast, Southeast, etc as well as just N, S, E, or W. If the brushes closed just one circuit, I knew that the wind was coming from one of the 4 main compass points. However, if the circuit repesenting East was closed, and at the same time, the North circuit was closed too, I knew that the wind was coming from the Northeast. I spent a lot of time trying to perfect this method, but I ran into enormous problems.

First, the brushes and contacts were exposed to the weather. I live in an area where rain can reach the acidity of orange juice. Any metal objects left in the elements corrode terribly. Such corrosion on the brushes would make it stop working after every storm. Also, the alignment of the brushes and the contacts must be very precise. Both the tension of the brushes and the location of the contacts had to fit within very close tolerances. While it was possible to build the assembly if great care was exercised, a strong gust of wind could knock the delicate arrangement out of alignment. Another disadvantage which I was not pleased with was the fact that each of the four contacts would require one bit of the input port. On the face of it, this doesn't seem like a big sacrifice. But four bits is four bits. Someday, we might wish we had them to use for something else. As I worked on one failure after another following the "brush/contact" tack, I kept thinking there had to be a better way.

What finally emerged was a completely different approach which eliminates the delicate, tedious to build brushes entirely. Instead, a variable resistor, the same thing used as the volume control in a TV set or radio, is altered slightly so it can be turned a full 360 degrees. The wind vane is mounted onto the shaft where the knob used to go. When this assembly is mounted out on a mast, the wind will turn the resistor's "wiper" resulting in a unique resistance which varies as the wind direction changes. What is more it is very easy to mount and still provide protection from the elements.

The varying electrical resistance produced by the wind is fed into an oscillator circuit. This produces a series of digital pulses whose frequency varies as the resistance varies. Therefore, in exactly the same way that digital pulses were read in last issue's anemometer circuit to determine relative wind velocity, the digital pulses produced by the oscillator, and governed by the wind vane, are read by the computer to determine wind direction. In fact, only four bytes of additional machine code are needed to make the wind speed software from the last issue also work to determine wind direction! And to make it even better, only one precious bit of the I/O Port is used, so we have oodles of unused bits waiting for future projects.

Now the Specifics

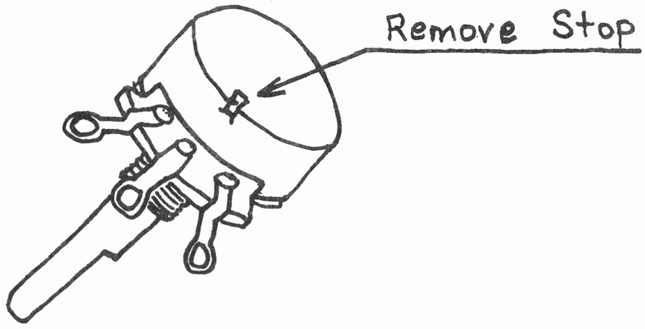
Phase 1 of building the computerized weathervane consists of obtaining a fairly high resistance variable resistor, also called a potentiometer, and performing some minor surgery on it so its shaft can turn a full 360 degree circle. You can pull one of these pots off an old TV or radio, or you can buy a new one from your friendly local Radio Shack dealer. What you want is one with a fairly long metal shaft, and one which has a total

Computer Updates is published quarterly in the winter, spring, summer and fall. Subscription price is $12.95 per year. In Canada and Mexico please add $3 postage. All other foreign subscriptions please add $9 postage. Back issues are available as single volume, 4 issue sets. Volume 1 is $9.95, volume 2 is $9.95

Edited and published by:
Thomas B. Woods
P.O. Box 64, Jefferson, NH 03583
(603) 586-7734

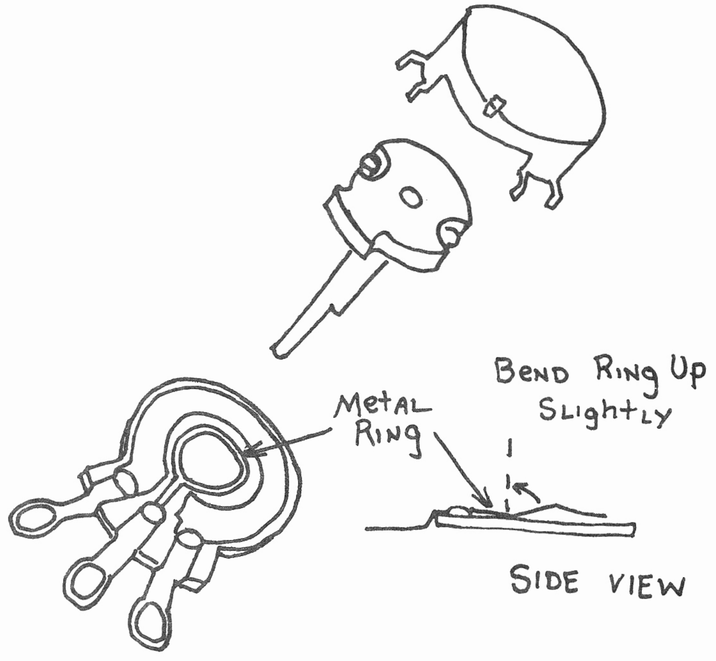
resistance of 500 Kohms to 1 megohm. The exact value is not critical. The resistor I used had a resistance of 679 Kohms which was measured between the two outside pins of the resistor.

Before you open the resistor up, look at the metal housing and notice that directly behind the center pin, there is a small slot where the case was punched inward (see diagram).



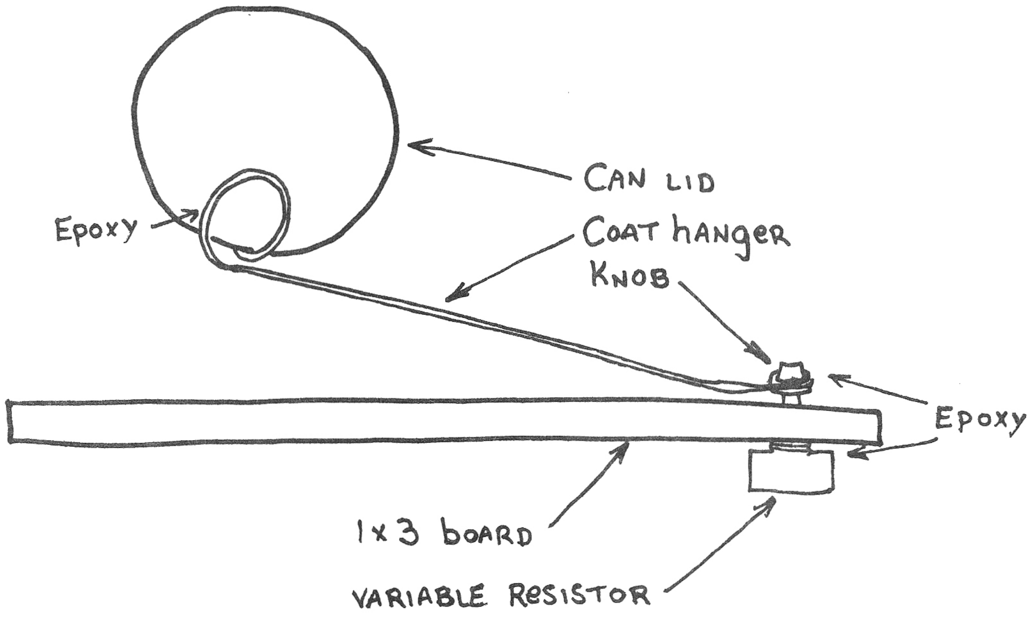
This is a stop which the innards of the resistor hit against if you try to turn the shaft too far. Since our purposes require that the resistor be able to turn in a full circle, we must remove this stop. To do it, use a small screw driver to bend back the metal tabs which hold the cover on. Then carefully remove it. With a hammer, punch, screw driver, pliers, and whatever else is necessary, beat away at the stop from the inside until you have either flattened it or removed it entirely. It doesn't really matter how you do it so long as the stop is removed to the point where the inside wiper can freely turn past it when the cover is put back on. I had very good luck just squeezing with heavy pliers until the stop was flattened.

While the variable resistor is apart, you can see that it consists of a rotor assembly with 2 brushes which slide along a black carbon material. This carbon is what creates the electrical resistance. Each end of the carbon is connected to an outside lug of the three connectors to the resistor while the center pin is hooked to a small metal ring which hits the rotor. If you try to turn the rotor at this point, you'll see that the brushes on the rotor get stuck on this ring as they pass it. Therefore, it is necessary to try to move the obstructing arm out of the way. It is very easy to do. Pull the rotor completely out of its hole so that the ring is in plain sight. Then with a screw driver, press down on the arm leading to the center lug right where the arm joins the ring. With your fingers or needle nose pliers, grasp the ring and bend it upward slightly so that you bend the arm right where the screw driver is holding it. Increasing the bend at this point will cause the arm to be squeezed down lower (and out of the way of the brushes) when the pot is reassembled. Put everything back together now and try turning the shaft a full 360 degrees. You will feel a slight drag as the brushes on the rotor cross the arm. This is unavoidable. What you should aim for is the smoothest turn past the arm you can get. If you're not satisfied, try bending the ring a tiny bit more.



Phase 2: Building the Weathervane

Now find a piece of narrow board such a 1 x 3 about 2 feet long. At one end drill a 3/8 inch diameter hole and turn the threaded part of the modified variable resistor into it. You may also want to gob a small amount of epoxy onto the resistor where it contacts the wood. This will prevent the weathervane from flying apart in a gale. The metal shaft should extend through the hole and out the other side of the board. Place the plastic knob back on the end of the shaft, and leave plenty of space so it does not drag on the wood. Now cut up the coat hanger and fashion it so one end fits tightly around the knob without slipping. Exactly how you do this depends a lot on the shape of the knob. If the one you use is round and offers



nothing to "grab onto" so as to prevent the wire from slipping, you might want to drill a hole in the knob to pass the coat hanger wire through. Be inventive. If all else fails, use more epoxy to hold it tight. Bend the other end of the coat hanger paper clip fashion so that you can slip the coffee can lid down into it. Make sure you have a tight fit. Later, after everything tests out ok, you will want to epoxy this in place as well, but don't do it yet. You may need to replace the vane with something larger to make it more sensitive to small puffs of wind.

At this point, the weathervane is finished. It should look something like the picture shown here. Different circumstances, however, will result in slightly different appearances. Do not be afraid to release your own creative genious on your own design. At any rate, the vane must now be set out in a light breeze to see how easily it turns. Ideally, the vane should turn even in a very light breeze. If it does not, check the resistor shaft and the coat hanger wire. Free them if they are dragging against the wood base. Another way to increase the vane's sensitivity is to increase the area of the vane. Use a larger piece of metal if necessary, or lenghten the distance to the resistor by using a longer piece of coat hanger. Remember, however, that the more weight you add, the more drag there will be, so use the lightest materials you can find. Once you're happy with the assembly, glue or screw the vane securely to the coat hanger.

Phase 3: Building the Oscillator Circuit

This part covers construction of the electronic circuitry which generates the digital pulses which are read by the computer. The schematic diagram shows how to wire up the very common 555 timer chip so it will output a frequency which varies as the variable resistor turns in the wind. To connect this resistor to the chip, use a long length of light weight speaker wire. This can be obtained in spools from Radio Shack or a HI-FI dealer. Solder one lead to the center lug of the resistor, and the other lead to either one of the other two lugs. It doesn't matter which. Power the 555 chip using the 5 volts supplied by your computer. If you build this circuit on the same board as the LM339 circuit used by the anemometer, you can simply extend the +5vdc and ground connections over to the 555. The output of the timer comes from pin 3. It is this pin which you want to connect to D1 of the Input port.

To check for proper operation and also to help understand what this circuitry does, wire everything up and temporarily connect an earphone between pin 3 and the ground. You will be able to actually hear the tone produced by the 555. By moving the weathervane to a different position, you'll hear the tone change. This is the signal which is fed into the computer through the input port. The program which reads the port, determines the frequency and prints the corresponding direction on the screen.

Phase 4: Now the Software

With the timer built, the weathervane constructed, and everything wired up to the computer, enter or load the program listing for the anemometer given in Updates Vol. 3, No. 1 (page 10, or 12 depending on which computer you use). Once its in, alter line 10 so it appears exactly as it does below. This alteration, makes it possible to use the same machine code for determining both wind speed and wind direction.

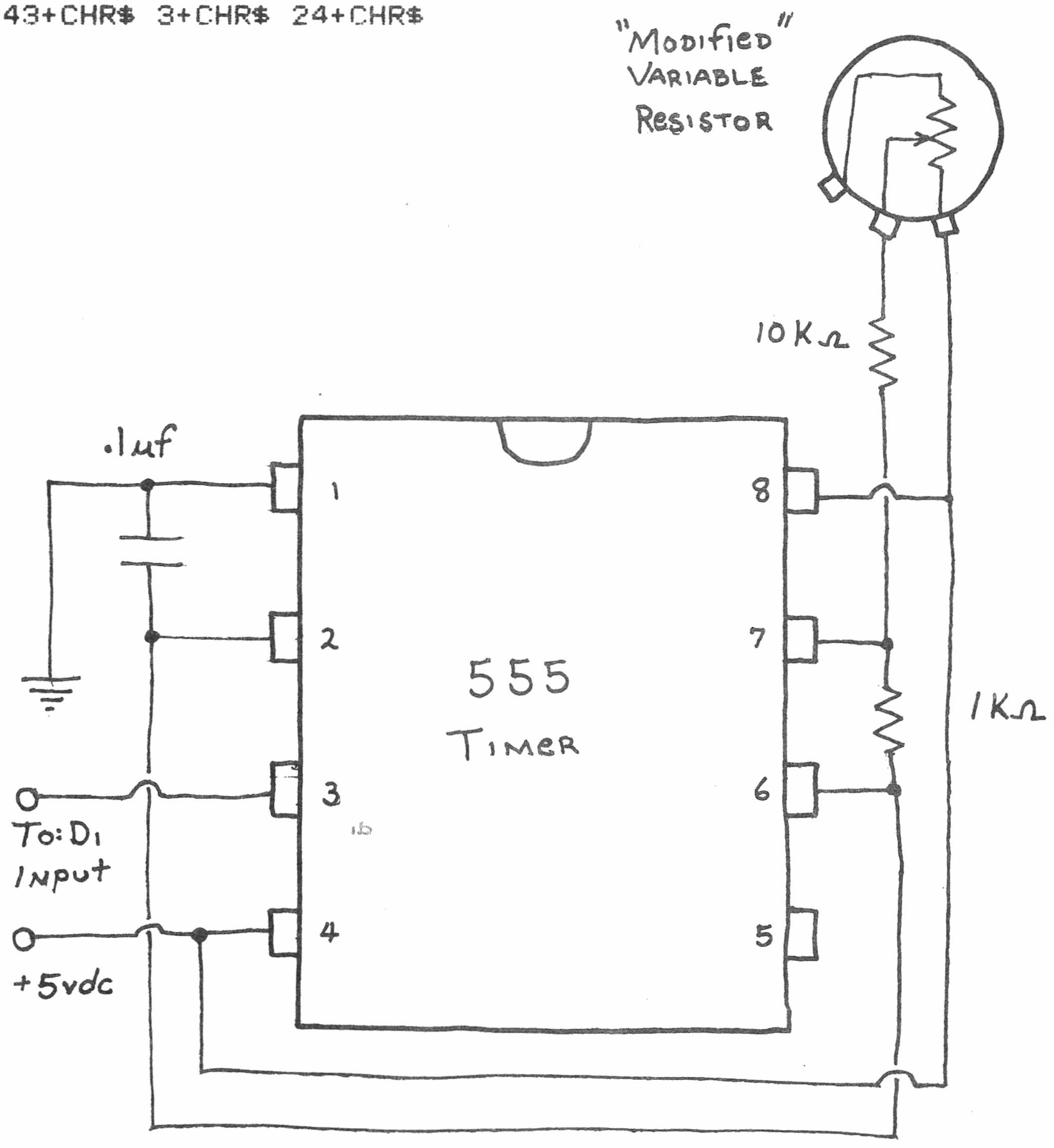
For TS2068

```
  10 LET a$=CHR$ 1+CHR$ 0+CHR$ 0
+CHR$ 118+CHR$ 62+CHR$ 195+CHR$
50+CHR$ 120+CHR$ 92+CHR$ 219+CHR
$ 207+CHR$ 230+CHR$ 1+CHR$ 95+CH
R$ 58+CHR$ 120+CHR$ 92+CHR$ 254+
CHR$ 0+CHR$ 200+CHR$ 219+CHR$ 20
7+CHR$ 230+CHR$ 1+CHR$ 187+CHR$
40+CHR$ 243+CHR$ 3+CHR$ 24+CHR$
239
```

For TS1000

```
  10 LET A$=CHR$ 1+CHR$ 0+CHR$ 0
+CHR$ 118+CHR$ 62+CHR$ 60+CHR$ 5
0+CHR$ 52+CHR$ 64+CHR$ 219+CHR$
223+CHR$ 230+CHR$ 1+CHR$ 95+CHR$
 58+CHR$ 52+CHR$ 64+CHR$ 254+CHR
$ 0+CHR$ 200+CHR$ 219+CHR$ 223+C
HR$ 230+CHR$ 1+CHR$ 187+CHR$ 40+
CHR$ 243+CHR$ 3+CHR$ 24+CHR$ 239
```

Add the lines on the next page to the Basic. They translate the frequency received into the more meaningful points of the compass and have been written so as to work equally well in both the TS1000 and 2068.



```
  95 IF X>64 THEN  LET X=64
 115 GO SUB 6000
 116 PRINT AT 6,0;"DIRECTION:";D
$;"    "
 117 PRINT AT 7,0;DIR;"    "
6000 REM DIRECTION SUBROUTINE
6002 LET MASK=2
6004 LET A$(13)=CHR$ MASK
6006 LET A$(24)=CHR$ MASK
6010 LET DIR=USR WIND
6011 LET MASK=1
6012 LET A$(13)=CHR$ MASK
6014 LET A$(24)=CHR$ MASK
6015 IF DIR<=58 THEN  GO TO 6100
6020 IF DIR<=65 THEN  GO TO 6110
6025 IF DIR<=85 THEN  GO TO 6120
6030 IF DIR<=105 THEN  GO TO 613
0
6035 IF DIR<=155 THEN  GO TO 614
0
6040 IF DIR<=225 THEN  GO TO 615
0
6045 IF DIR<=400 THEN  GO TO 616
0
6050 LET D$="NW"
6055 RETURN
6100 LET D$="N"
6105 RETURN
6110 LET D$="NE"
6115 RETURN
6120 LET D$="E"
6125 RETURN
6130 LET D$="SE"
6135 RETURN
6140 LET D$="S"
6145 RETURN
6150 LET D$="SW"
6155 RETURN
6160 LET D$="W"
6165 RETURN
```

How Does the Software Work?

You will recall from the last issue that the function of the short machine code, USR WIND, is to monitor the input port and and determine how many transitions from low to high occur in one second's time. In other words, it determines frequency. The four previously mentioned bytes which are added this time do not alter this. Their job is to allow the computer to look at the pulses coming in at just one individual bit of the port at a time. In this way, we can have a frequency coming in at bit 0 from the spinning anemometer, and another frequency at bit 1 from the weathervane. The computer can use the same machine code to monitor both, and we still have 6 more bits of the port for future uses.

The disassembly here shows USR WIND. If you compare it with that found in the last issue of Updates, you'll see that the difference is that after every IN instruction, there is inserted an AND 01.

```
BGIN LD BC,0000
     HALT
     LD A,3C
     LD (FRAM),A
     IN A,(DF)
     AND 01
STOR LD E,A
CNT_ LD A,(FRAM)
     CP 00
     RET Z
SMPL IN A,(DF)
     AND 01
     CP E
     JR Z,CNT_
     INC BC
     JR STOR
```

Using AND in this way is a very useful technique for "masking", or ignoring, unwanted bits of a number. What happens with the AND instruction is this: The computer matches each bit in the accumulator with the corresponding bit of the number following the AND instruction. If both bits are 1, the computer keeps the 1 in the given bit of the accumulator. But if one of the bits is 0, the computer loads a 0 into that bit of the accumulator.

Therefore, we can input a value from the port which may have any number of bits set to 1. If we don't care about the state of any of the bits except bit 0, we could use AND 01 to turn all the bits in the accumulator to 0 except the right most bit. If we want to look only at bit 1 (the next bit to the left), we could use AND 2 which in binary is 00000010. Here, every bit will be reduced to 0 except bit 1.

Quiz time: True/false- You would use the AND 03 instruction to mask all bits except bit 2 (the third from the right.)

Answer: False. AND 03 or binary 00000011 would mask all bits except bits 0 and 1. Bit 3 would be forced to 0.

When the computer is determining wind velocity, it wants to look at bit 0 so AND 01 is used. When it determines wind direction, it wants to ignore bit 0 and look only at bit 1 since that is where we are feeding the tone. Lines 6002 to 6006 change the AND 01 into AND 02 to accomplish this. It is the 13th and 24th characters of A$ which store the number to be ANDED with the accumulator. We simply use LET statements to alter the machine code in A$ accordingly.

Note that lines 6011 to 6012 change A$ back to the original AND 01 instruction so that when the subroutine returns, the computer will look at the right bit when it determines wind speed again.

Converting a Tone Into a Compass Point

After the computer obtains the frequency produced by the timer (lines 6000 to 6014), it must next translate that frequency into a point on a compass. This is done by lines 6015 to 6045 in a straightforward manner. Line 6015 says:

```
    IF DIR<=58 THEN GOTO 6100
```

In English, this means, "If the tone has a freqency of 58 or less, execute the program lines starting at 6100. This is where the computer assigns the first possible compass point, "N" or "North" to D$.

If the frequency is greater than 58, line 6020 considers the next possible frequency range and if it falls between 59 and 65, D$ is assigned the value "NE". The computer simply plays the game, "if you see this, print that" when it decides which direction to assign to D$. "If the tone is within this range, make D$ equal that compass point."

The values shown in lines 6015 to 6045, of 58, 65, 85, 105, 155, 225, and 400 are values I obtained when I calibrated my weathervane. They will not be the same for you because the the components you use in the 555 timer circuit as well as the way you position your weathervane can result in a totally different frequency. Through trial and error, you must determine the proper values for these program lines based on the way your weathervane is configured.

This calibration can be done fairly easily. Before you actually mount the weathervane high on a mast, set it within easy reach, but point the board in the same direction it will go when it is fastened to the mast. Using a compass, mark out the 8 compass points on the board, and then turn the vane arm so it is pointing North by Northeast. Run the program and note the number which is printed just below the line which says "DIRECTION:". This number is the value you want to stick in line 6015 in place of the 58 that I used.

Next, swing the arm so it points East by Northeast and run the program again. The number printed on the screen replaces the 65 I used in 6020. Again, point the vane to East by Southeast. Run the program and use the number displayed to replace the 85 in line 6025. By going through the remainder of the compass points, you can fill in the proper values for 6030 to 6045.

On the surface, this calibration process seems very simple, but two complications can arise which can prove to be troublesome. They both have to do with the nature of the modified variable resistor used to govern the tone frequency. If you hook up an earphone and listen to the tone change as you sweep the vane around in a circle, you will hear the tone gradually increase until it reaches a maximum. Then, as you continue the sweep, the tone instantly drops to the lowest frequency and gradually begins its upward climb again. Depending on the alignment of the vane assembly, it is possible that the lowest frequency and the highest frequency both represent the same compass point. If this happens to you, there are two things you can do: 1. re-align the vane assembly to eliminate this situation, or 2. program the computer to handle it. I chose option 1 but programming around the problem is not too difficult. You would simply add to the affected program line like so:

```
    IF DIR > nn OR DIR<= xx THEN GOTO yy
```

In this line "nn" would be some value just less than the maximum tone. "xx" would be the low frequency cut-off point, and "yy" is the proper program line to go to.

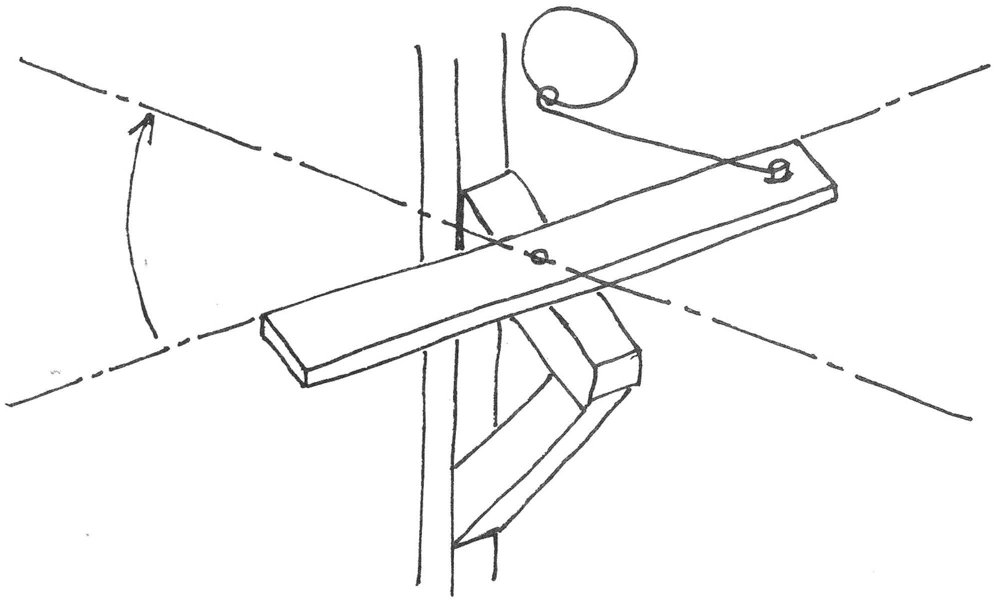
The second complication which can crop up stems from the fact that our "modified" resistor is being made to do something it was never intended to do. That is, swing around in a full circle. I suspect that different resistors will produce different results, but every one that I used had an area where the vane could turn without changing the tone frequency. In a "bad" resistor, this area could cover an arc of as much as 45 degrees. "Better" resistors were substantially less. Again, depending on the alignment of the entire weathervane, you could find that this "dead area" coincides precisely with a major compass point. The only solution I could come up with was to carefully align the vane so that the dead area was between compass points rather than smack dab on top of one. For most hobbyists, aligning the vane will not pose too much of a problem. But, those of you who really get into this project and want to improve "resolution" to much more precise compass points beyond the 8 major directions, will want to find a variable resistor capable of rotating the full 360 degrees. I understand that such resistors are manufactured, but they are not common and you'll have to dig deeper than a Radio Shack store to find it. If anyone finds a source for these devices, please let me know and I'll pass the word along.

Final Points

After you have written the software, built the hardware, and worked out all the bugs, congratulate yourself for being so tenacious.

The final phase has come. Now it is time to mount the weathervane on the mast along with the anemometer you built. Use angle brackets and screws to position the weathervane just below the wind cups, bearing in mind that you must keep the board which supports the resistor and vane pointing in the same direction that it was when you calibrated the system. One very good way to insure that this will be the case is to first place your compass on the board and note its degree heading during the calibration process. Then, mount the weather vane on supports as illustrated using just one screw to hold it in place. Put the compass back on the board and rotate the assembly until it is in exactly the same position it was for calibration. Then fasten the board tight using more screws. Be sure that the vane arm will clear the mast as it twists in the breeze.

Now you're fully outfitted with a digital computerized recording anemometer and wind direction indicator. You could have bought one of equal value for $300 to $600. Instead you got by with a $70 port board, a cheap computer, and about $1200 worth of your time. But it was great fun wasn't it? Stay tuned to future Updates articles which will add more weather monitoring capabilities. If you think of something to add yourself, let us all know about it.



Rotate assembly to proper heading.
Then fasten with more screws.

BIG ADVANCE IN PROGRAMMING THE 32K NVM

Here's some good news for 32K Non-Volatile Memory owners. Now there is a way to store your Basic programs in the Dock bank on the TS2068 and be able to edit them directly without transfering them into the Home bank first. This makes writing and debugging AROS programs as easy as regular Home bank programs.

With an empty computer, an empty 32K board plugged in, and the write protect switch set to WR, enter and run the program lines below:

```
  10 OUT 244,240: FOR x=32768 TO
 32777: READ y: POKE x,y: NEXT x
: DATA 1,2,10,128,15,0,0,0,27,13
1
  20 LET y=32778: FOR x=26710 TO
 27710: LET n=PEEK x: POKE y,n:
LET y=y+1: NEXT x
  30 POKE 33563,128: POKE 33564,
13: POKE 33565,128
9997 STOP
9998 CLEAR : POKE 23750,0: OUT 2
44,240: POKE 23635,10: POKE 2363
6,128: POKE 23627,PEEK 32776: PO
KE 23628,PEEK 32777: RANDOMIZE 1
+PEEK 23627+256*PEEK 23628: POKE
 23641,PEEK 23670: POKE 23642,PE
EK 23671: STOP : REM edit aros
9999 CLEAR : POKE 32776,PEEK 236
27: POKE 32777,PEEK 23628: POKE
PEEK 23627+256*PEEK 23628,128: P
OKE 1+PEEK 23628+256*PEEK 23628,
13: POKE 2+PEEK 23627+256*PEEK 2
3628,128: REM save pointer
```

Now set the write protect switch to PR and reset the computer by turning it off, then back on. This program, when run, transfers to the dock bank. Lines 9998 and 9999 allow you to directly edit in the dock bank.

To produce the listing, enter the command, RUN 9998. After you get the OK prompt, press enter a second time. The listing will come up on the screen. After you see the listing, set the write protect switch to WR, and you can add or edit program lines.

First, you will want to delete everything except lines 9997, 9998, and 9999 because they are no longer needed. Then try entering a few simple lines for testing purposes such as:

```
10 PRINT "HELLO"
20 GO TO 10
```

Once this is done, enter the command RUN 9999, and after you get the "OK" prompt, set your write protect switch to PR. Turn off the computer and then turn it back on to reset the machine. The computer will now run your AROS program when you type RUN and ENTER. To list and edit the AROS, enter the command:

```
RUN 9998
```

When you do, the computer will execute line 9998 and return you to the OK prompt. As before, press just ENTER a second time. The AROS listing will come on the screen. When you see the listing come up, you know it is safe to set the write protect switch to WR so that you can add or edit new program lines. You will be working directly in the Dock Bank!

After every edit session, enter RUN 9999, and immediately thereafter, before you even breathe on your computer, set the write protect switch to PR.

Before every edit session, enter RUN 9998. After the computer reports "OK", press ENTER again to produce the listing. After the listing comes up on the screen, set the write protect switch to WR and you're ready to edit.

When Timex engineered the TS2068, they probably never dreamed that anyone would be using RAM in the dock bank, much less editing Basic in the Dock. Therefore, a few shortcomings in the ROM must be avoided or you run the risk of crashing the computer. I use the term "shortcomings" rather than "bugs" because we are doing something with the computer which Timex never designed into the machine. The problems involve trying to add or edit any program line which contains the SAVE, LOAD, VERIFY, or MERGE commands, and stem from the way the computer checks the syntax of a line.

This in itself, could be the subject of a lengthy article, but briefly what happens is this: When you try to enter a program line, either a new one or an old one which you have changed, the computer actually begins to run the line as if it were a direct command without a line number. It does this to see if an error will be generated. If it finds something in the line which is incorrect, the machine refuses to put the line in the program. Instead, it prints the line once again at the bottom of the screen and sticks a flashing question mark (syntax error) at the point where if found the mistake. The flashing question mark tells you how far it got in parsing out the line. The problem with SAVE, LOAD, VERIFY, and MERGE, is that the computer ALWAYS assumes that these tape commands will work on bytes in the Home bank. But we are running in the Dock bank! When the computer trys to run the line to check its syntax, it enables the Home bank and does not turn the Dock bank back on when it is finished checking the line for errors. Therefore, regardless of whether the line is syntactically correct or not, the computer literally locks you out of the dock bank where you should be. Instead, you are stuck in the Home bank with a whole bunch of garbage on the screen and no way to get back. The lesson is:

NEVER, NEVER, NEVER try to add or edit a program line which contains a tape command when you are editing directly in the Dock bank unless, of course, you want to commit suicide on your program. Fortunately, ALL other basic commands work.

Ok, I can hear you grumbling, "What good is it if I can't save or load from an AROS program?" The answer is there's nothing which prevents you from saving or loading from AROS. The restriction is that you can't enter a program line which says SAVE, LOAD, etc. There is a sneaky way around the problem albeit somewhat combersome. What we must do is enter a different command that WILL pass through the syntax check, and then Poke the tape command in manually.

Suppose you want to enter the program line:

```
125 LOAD "PROGRAM"
```

Because the system will crash if we try entering this, replace it with

```
125 PRINT "PROGRAM"
```

As you can see, the two lines are almost identical; the only difference being the substitution of the PRINT token for LOAD. It is possible to write a short Basic program which can peek into the NVM so we can find line 125. Once we have the address of the PRINT token, we can simply POKE it with the code for the SAVE token. Before you try the following listing, enter the command RUN 9999 and then set the write protect switch to PR. Remember, it is absolutely essential that you do this EVERY time you finish an aros editing session. Once its done, reset the computer, and POKE 23750,0 to turn OFF the AROS thereby allowing you to write this program in the home bank:

```
 1 OUT 244,240
10 FOR X=32778 TO 65535
20 PRINT X;"=";PEEK X;TAB 12;
30 IF PEEK X>=32 THEN  PRINT
   CHR$ PEEK X: NEXT X
40 PRINT : NEXT X
```

If you run this program, the screen will fill with something like that which appears below. It shows the bytes of memory in the dock bank along with the peek values and character codes. Do you see the Basic program? The first two bytes represent the first program line (line 125). The next two are the length of the line (11 bytes long). The next 10 bytes are the text of the line (PRINT "PROGRAM"), and address 32792 shows the code 13 which is the carriage return at the end of the line.

```
32778=0
32779=125    }
32780=11
32781=0
32782=245     PRINT
32783=34     "
32784=80     P
32785=82     R
32786=79     O
32787=71     G
32788=82     R
32789=65     A
32790=77     M
32791=34     "
32792=13
32793=39     '
32794=13
32795=2
32796=0
32797=226     STOP
32798=13
```

Note the PRINT token at address 32782. This is the byte that needs to be changed to say SAVE. To do this, set the write protect switch on the NVM to WR and look up the code for SAVE on page 245 of the 2068 owner's manual. You'll find there, that the decimal code for SAVE is 248, so just POKE 32782,248. Then write protect the memory board again.

The next time you go into the dock edit mode (using RUN 9998), you will find that the old line which said PRINT "PROGRAM" now says SAVE "PROGRAM" and if you run this line as an aros program, the contents of the home bank will be saved to tape.

In this example, I conveniently placed the line which needed to be poked right at the start. In YOUR program, the save command may be located somewhere else and you will have to hunt for the line to find the right address to poke.

Conclusion and Summary

This article just scratches the surface of the very deep subject of using the 32K Non-Volatile memory. I will be writing much more about it as I learn its intricacies myself. This article does help a lot in making the programming process easier, but using lines 9998 and 9999 can be confusing. With lots of practice, you'll become quite adept. Now, here is a summary of the purpose of lines 9998 and 9999 along with some Do's and Don'ts.

After you have put lines 9998 and 9999 in the NVM, you can list and edit the aros basic by entering RUN 9998. Execute this command with the write protect switch set to PR. Only after you actually see the listing on the screen is it safe to move it to the WR mode.

Think of the RUN 9998 command as opening the door to your aros program. You can't look inside until you have opened this door.

Once its open you can add new program lines, delete, or alter existing ones. But any time you decide to turn off the computer, stop, or take a short break, you must close the door to the aros. This is done by executing the command RUN 9999 and then, immediately after, switching the write protect switch to PR. If you do not RUN 9999, all your work will be lost!

Do not try to run the program you're working on if you have opened the door to edit the aros. Results are unpredictable. Instead, run 9999 first. Write protect the memory, reset the computer, and type RUN to run it as an aros program. The only line which is safe to run while editing aros Basic is line 9999 which closes the door.

Do not try to edit or add any program line which contains one of the tape commands (SAVE, LOAD, VERIFY, MERGE). The computer will crash if you do. Instead, insert one of these "killer tokens" indirectly by writing the line using some other token, and then substituting it with SAVE, LOAD, etc. by peeking into the Basic, and Poking in the substitute at the appropriate address.

Remain constantly aware of which way you have the write protect switch set. This is good practice any time you use the NVM, but especially so when you edit an aros program. If you're not writing to dock memory, keep the board protected to avoid accidents.

TS1000/1500 EXTENDED BASIC SCREEN, DRAW, and FILL DEMO

Here's a program written in Extended Basic to demonstrate the use of the new commands DRAW, FILL, and SCREEN. For those not familiar with Extended Basic, this program adds 22 new Basic commands to the existing TS1000 vocabulary. The three used in this program are from a large group of "display commands". DRAW is used to draw a line between two points on the screen. The effect is similar to drawing a line using the Sinclair PLOT command, only

```
   1 ~REM
   2 LET A=10
   3 LET M=23
   4 LET B=10
   5 LET S=0
   6 LET C=5
   7 PRINT AT 18,7;"HIDDEN EXPLO
SIVES"
   8 LET D=5
  10 LET X=0
  11 GOSUB 0
  12 REM DRAW;A,B,C,D,
  15 GOTO 1005
  20 GOSUB 0
  21 REM DRAW;A,B,C,D,:SCREEN;16
,12,X,
  30 IF X THEN GOTO 2000
1005 LET A=C
1010 LET B=D
1020 LET C=1+(INT (RND*63))
1030 LET D=1+(INT (RND*43))
1040 GOTO 20
2000 FOR X=1 TO 5
2004 GOSUB 0
2005 REM FILL ;11,13,15,17,(M),11
,13,15,17,(S),
2007 NEXT X
2010 GOTO 1005
```

draw is much much faster, and easier to use when making diagonal lines. SCREEN is a function, which returns the character of a given column/line position of the TV display. It tells you, for example, what character is printed at column 6, line 4. The TS2068 has both of these commands and they work in almost exactly the same way. The third command demonstrated here, FILL, is unique to Extended Basic. You won't find anything like it on anything less than the new QL computer. FILL lets you fill an area of the screen of whatever size you specify with any desired character or graphic. And its F-A-S-T. The area literally fills instantly. This makes the command particularly useful for explosions, flashing screens, window simulations, etc.

When you study the listing, you will see that it looks pretty much like any other Sinclair Basic except for lines 11-12, 20-21 and 2004-2005. These lines are the three places where the Extended Basic is invoked.

What this program does is draw a random path across the screen in a zig-zag fashion. Hidden from view is a box of explosives. If the line passes over the bomb, an explosion occurs. The drawing of the random path, the check to see if an explosion is needed, and the explosion itself are all accomplished with Extended Basic. Everything else is done in Sinclairese.

HOT Z CLINIC

Hot-Z for both the TS1000 and 2068 has been in constant demand since I first first added it to my product line. Make no mistake about it, this product is a very powerful tool in the hands of a machine code programmer. Unfortunately, there is another side to the Hot-Z coin. Right from the start, customers complained that they couldn't figure out how to use Hot-Z. Some said that the documentation (about 50 pages) went right over their heads.

HOT-Z CLINIC is going to be a regular column in Computer Updates. It is for all owners of the program who have experienced difficulties. And that includes everybody, even me. I must confess that even after 2 years of using Hot-Z, there are still some features I have not dared to try. In each column I will concentrate on just one or two features, using very short and simple examples. Hopefully, I can pick up where Hot-Z's documentation leaves off, and provide that one single comment that makes everything click into place for you.

So what is all this fuss about Hot-Z, anyway? Above all else, Hot-Z is a tool for writing machine code. It is to the programmer what a table saw is to the fine finish carpenter. Anyone can use a table saw, but the skilled craftsman can use it to produce items the weekend nail pounder can only dream of. Hot-Z will NOT write machine code for you. It will NOT magically turn you into a hot shot programmer. You plug it in, turn it on, and use YOUR creative skills to write. Like the saw, Hot-Z sure beats cutting it by hand. If you know how to use it, the program will speed your production, and enable you to do things you'd never attempt without it.

Now, an apprentice carpenter does not start by building a beautiful set of custom cabinets. Nor should you as a Hot-Z beginner expect to churn out kilobytes of poetic machine code. You start with the Basics. So with that in mind, here is a demonstration of two of the most fundamental rudimentary functions Hot-Z has to offer: reading a disassembly and switching between a data display and a disassembly.

First, load your copy of Hot-Z into the computer. If you use Hot-Z II for the TS1000, the load will finish with a title screen and a prompt which tells you to press any key to begin. Hot-Z AROS, the cartridge verion for the 2068 also comes up with a title screen and asks you which kind of printer interface you use. With these two versions, enter the information asked for. After you do the first screen of Hot-Z's disassembly display will appear. This same display comes up automatically after loading the tape version of Hot-Z for the 2068. What you'll see is something like that shown below.

```
Addr Hexcode Label Mnemonic
============Bank=FFro=====
0000 F3           DI
0001 AF           XOR A
0002 11FFFF       LD DE,FFFF
0005 C3310D       JP init
0008 2A5D5C       LD HL,(chad)
000B 225F5C       LD (xptr),HL
000E 1843         JR ertr
0010 C3ED11       JP pra2
0013 FF           RST 38H
0014 FF           RST 38H
0015 FF           RST 38H
0016 FF           RST 38H
0017 FF           RST 38H
0018 2A5D5C       LD HL,(chad)
001B 7E           LD A,(HL)
001C CD7D00       CALL skip
001F D0           RET NC
0020 CD7400       CALL nxch
0023 18F7         JR 001C
0025 FF           RST 38H
0026 FF           RST 38H
0027 FF           RST 38H
```

Hot-Z always starts up in what is called the READ mode. What you see on the screen is a disassembly of the first 22 instructions of your computer's ROM. Hot-Z is "reading" the numbers stored in memory starting at address 0 and displaying what it finds in both the Hexadecimal representation and the mnemonic.

The far left column of four digit numbers, starting with "0000" and going to "0027" at the bottom of the screen, represent the starting memory addresses for each Z80 machine code instruction.

The next column, labeled "Hexcode" is the hexadecimal representation of each instruction. The number of bytes needed to write any given machine language instruction varies depending on the instruction. In this display, several are shown which take up only one byte such as those at addresses 0000 or 0001. Longer instructions which require three bytes appear from addresses 0002 to 000B. Next, a two byte instruction is shown at 000E.

The top number in this column, the F3, is a hexadecimal display of the value stored in the very first byte of the ROM. The hex numbering system is used throughout the program for a variety of reasons. One of the bigger ones is that of convenience. A number stored in any single byte of memory can range in value from 0 to 255. To display values in decimal would, therefore require from 1 to 3 digits. Hex is used because the same values can be expressed in just 2 digits (00 to FF) per byte. This makes the display more compact and also eliminates a great deal of confusion which can arise when decimal numbers are used. Consider the instruction at address 000E. We can tell by looking at the hexcode, that this is a two byte instruction comprised of 18 and 43. If this were a decimal display, would the values be 18 and 43? Or could they be 184 and 3? Hot-Z follows the convention of hex notation and always uses 2 digits to represent the value of each byte. Therefore, in a longer instruction such as at address 0002, we have a 3 byte instruction which is written in hex as 11, FF, and FF.

Another reason why Hex is used is that it is (believe it or not) easier to understand. The explanation of why this is so also makes a nice introduction to the third column of the TV display, the Mnemonic representation of each machine language instruction. Don't let the word "mnemonic" throw you. I can't pronounce it either. What it IS is more important. The mnemonic display is an abbreviated English (?) rendition of the hexcode instructions in column 2. Once you learn how to pronounce these abbreviated words, it will much easier to understand what a piece of machine code does. Literally thousands of pages have been written about machine code including great volumes on exactly what happens inside the computer when some specific instruction is executed. I certainly do not intend to duplicate all that work here. But I will tell you that the mnemonic "DI" means "disable interrupts". That definitely holds more meaning than saying "F3".

Look at the mnemonic at address 0002 which says LD DE,FFFF. In English, this stands for LOAD DE,FFFF or "load the register called DE with the number FFFF." Now I can attempt to explain why working exclusively in hex is easier to understand than working in decimal. For your information, FFFF hex is the same as 65535 in decimal. If we were to write mnemonics in decimal, this instruction would become LD DE,65535. The hexcode translated to decimal would be 17, 255, 255. It is very difficult to look at the decimal numbers and see any correspondence between the two 255's and the mnemonic 65535. But when we deal exclusively in hex, the hexcode and the mnemonic are the same. We have two FF's in the hexcode column and FFFF in the mnemonic column as well. There are very definite advantages in this which will become more apparent as you progress.

So far, we have looked only at the first screenfull of disassembly. To view the next screenfull, press ENTER on the TS1000 or the SPACE BAR on the 2068. This will let you "page through" consecutively. There is another method which allows you to start the display at a specific address. This is done by simply typing a four digit hex address. To test this out, try typing 013E. After the last zero is pressed, the display will instantly show a new disassembly starting with address 013E hex.

Actually, this is an area in the ROM of both the 1000 and 2068 which stores the tokens used by the machine (like PRINT, GOTO, VAL, etc.). Because Hot-Z is in the disassembly mode, it takes the bytes it finds here and shows you the machine code mnemonics even though it is data rather than machine code. To the trained eye, disassemblies of data are fairly easy to spot because such a disassembly results in nonsense. This is not because Hot-Z is doing anything wrong. On the contrary, the program is doing a masterful job trying to make sense out of bytes which are not supposed to be machine code. Hot-Z has not been told that it is looking at the token table.

Built in to the program is the capability to read memory without trying to disassemble it. This is called the "DATA DISPLAY." To see the token table for what it really is, press SHIFT and the letter D on the TS1000, or Symbol Shift and G on the 2068. Once you are in the DATA mode, you see a completely different presentation of memory. As in the Disassembly mode, you have the hex address and hex value stored there. But you also get these numbers printed in decimal. The right hand column shows the character code for the value stored in each given byte. Can you see some tokens hiding there? On the TS1000, you should see SQR starting at address 013E. Following that are SGN, ABS, and PEEK. On the 2068 you should see MERGE starting at 0141 hex, followed by VERIFY and BEEP.

You can look at any area of memory in either the Data or the Disassembly mode. DATA mode is just like Disassembly mode. Press ENTER (for TS1000) or SPACE BAR (TS2068) to page from one screen to the next. Or, type in a four digit hex address to start a display at at the address you type. The key you pressed to switch from DISASSEMBLY to DATA is a switch to go back and forth. Press the same key again to go back to the Disassembly mode.

Here's a tip. How do you use Hot-Z to translate a hex number into decimal? Answer: Flip to the DATA mode and type the hex number you want to translate as if it were an address. The screen will jump to the address you type in. Since you are in the DATA mode, the decimal equivalent of the number you typed will appear in the forth column from the left.

Well, this Hot-Z Clinic will close for now. We covered quite thoroughly, the subject of READING. Next time, I'll delve into WRITING, and if there's time, maybe a little ARITHMETIC.

HOW TO USE THE ROMPAK TS1000 PRO/FILE

My old pal, Floyd Cox, writes, "How do I use the Rompak Quickload and Pro/File cartridge on my TS1000?"

Answer: The Quickload Pro/File cartridge which was manufactured by Rompak, and is still being sold by Sunset Electronics in San Francisco, plugs into the back of your computer. When you power the computer up, the "K" cursor appears as usual, but by entering the command RAND USR 9000, the ZX Pro/File menu comes up instantly, giving you an empty data base with a capacity of 10680 characters.

After adding information, you can save the program by typing SAVE at the main menu, however, unlike the cassette version, you are given the option of making a regular tape save, or a high speed Quicksave. If you select the Quicksave option, the entire Basic program and data will go out to the tape recorder at a rate 15 times faster than normal. Once a program has been quicksaved, you can then use an additional Quickload facility to load it back into the computer. Here's how:

First turn on the computer and wait for the cursor to appear. Next enter the command, RAND USR 8200. Play the tape which has the Quicksaved program on it. Assuming you use a good recorder and tape when you saved the program, you'll be able to load it with a minimum of loading errors. The Pro/File menu will come up in 30 seconds!

For some reason, many people have not been able to figure out how to use this cartridge. I am, quite frankly, perplexed. It works just fine for me in a very reliable, simple, straightforward way. Here are a few points about using the Rompak Pro/File to remember:

*Don't forget to plug in your rampack!
*The cartridge Pro/File will work only with the TS2040 printer.
*You can't Quickload a program which has been saved using the normal SAVE command.
*Quickload only works on programs which have been Quicksaved!

You don't have to have Pro/File in memory to take advantage of the Quicksave and Quickload features. ANY program which is first loaded into memory can be Quicksaved using the command RAND USR 8192, and then Quickloaded with RAND USR 8200.

If you have existing Pro/File tapes which were saved in the normal way, you can adapt them to Quickload very simply. Just load the tape using the normal LOAD command. When the main menu comes up on the screen, break into the listing and change line 25 so it says:

```
25 IF X$="SAVE" THEN RAND USR 8192
```

Go back into the Pro/File program using the GOTO 17 command, and type SAVE at the main menu. The program and all your data will now Quicksave and you can load it back by entering the command RAND USR 8200.